KOSTIUK D.A., CHETVIORKINA G.A., JUK A.M., SIDOROVICH T.P. Migration of the low-level programming and computer architecture courses to GNU/Linux

The specifics of GNU/Linux operating system usage to teach students low-level programming and computer architecture are investigated. Difficulties related to the platform differences are analyzed, as far as choice of the software and available information sources. Approaches to the arising problems are discussed based on practical experience of implementing the subject in academic activity.

УДК 681.3

Муравьев Г.Л., Хвещук В.И.

О ПОСТРОЕНИИ СИСТЕМ ОБУЧЕНИЯ КОНСТРУИРОВАНИЮ ПРОГРАММ

Введение. Правильно организованное обучение с использованием различных компьютерных сред дает опыт работы с общими принципами, методами разработки и программирования, позволяет адаптировать приобретенные навыки при освоении других средств разработки ПО. К настоящему времени сформулированы проблемы, принципы, требования к обучению в рассматриваемой области, отраженные в стандартах, программах изучения информатики. Это, например, программы ECDL по профессиональной сертификации навыков владения персональным компьютером и повышения компетентности специалистов в сфере ИТ, представленные на сайтах http://www.ecdl.org/programmes, http://www.ecdl.su, учебные планы по информатике ICF-2000, разработанные Международной федерацией по обработке информации IFIP под эгидой ЮНЕСКО и др. [1], где сформулированы такие направления, как обучение проектированию программ на подробно комментированных образцах, на принципах доказательного программирования, методом пошаговой разработки и др. Обучение конструированию программ предполагает выработку специальных навыков работы, включая спецификацию артефактов, проектирование архитектуры, спецификацию алгоритмов решаемых задач, кодирование на языках высокого уровня (ЯВУ), тестирование спецификаций, документирование и т.д. [2].

Таблица. Характеристики систем

№	Характеристика	вид системы			
		1	2	3	4
1	многооконность	+	+	-	+
2	выделение синтаксиса	+	*	-	+
3	средства структурирования текста	+	+	-	*
4	поддержка схем иерархии	+	-	-	-
5	средства спецификации	+	*	-	*
6	русифицированность	+	+	+	-
7	склоняемость имен	+	-	-	-
8	синонимы команд	+	-	-	-
9	пошаговая отладка	+	+	*	+
10	точки останова	+	-	-	+
11	наблюдение диаграмм переменных	*	-	-	-
12	создание exe-модуля	+	-	-	+
13	генерация текста на ЯВУ	+	-	-	-
14	библиотеки эталонных примеров	+	*	*	+
15	средства тестирования	+	-	-	-
16	средства верификации	*	-	-	-

Характеристика систем обучения. Наряду с широко используемыми в целях обучения системами программирования существует много специальных систем, автоматизирующих обучение.

Например, среды на базе языка Лого, системы программирования Комплекта Учебных МИРов (КуМир), тренажеры, web-визуализаторы тренажеров и др. Однако конструирование программ в части работы со спецификациями, архитектурой, модульного проектирования, прототипирования, генерации каркасов обучающими средами поддержаны слабо. Как правило, отсутствуют отладчики спецификаций, нет средств генерации текстов на языках высокого уровня (ЯВУ). Оценка корректности проектных решений, спецификаций, поиск и локализация ошибок, тесно связанные с обучением конструированию программ, разработке алгоритмов, как правило, производится вручную. Используемые лингвистические средства обеспечивают слабую документированность и читаемость кодов.

Результаты сравнительного анализа базовых характеристик систем, автоматизирующих обучение, представлены в таблице. Здесь в графе 1 приведены требования к характеристикам перспективной системы, в графе 2 – характеристики системы КуМИР, в графе 3 – характеристики типовой системы обучения, полученные по “усредненным” данным таких систем, как [3–7] и др., в графе 4 приведены характеристики типовой системы программирования. Это базовые характеристики средств редактирования и структурирования (1–5), возможностей комментирования (6–8), отладки (9–11), средств генерации (12, 13), тестирования (14–16). Символ ”+” означает наличие средства в системе, “*” – желательность средства для систем первого типа и частичность наличия в системах остальных типов. Системы 2–4 отличаются использованием данных средней типизацией, а для системы первого типа достаточно использование стандартных скалярных и структурных типов, в том числе имеющих математические аналоги. В системах 2–3 используются строчные, в 4 – строчные и фрагментарные комментарии, а в системе первого типа необходимы строчные, фрагментарные и ”встраиваемые” в текст комментарии. Исполнимость систем 2–3 основана на интерпретации спецификаций, что достаточно и для системы первого типа.

Структура и функционирование системы. В практике разработки ПО прототипирование является этапом получения пробных версий программ для оценки принимаемых решений и согласования их с заказчиком. Оно предусматривает: спецификацию требований, сценариев работы; разработку модульного каркаса проекта, прототипа интерфейса с акцентом на отработку общего управления; – изучение прототипа, тестирование спецификаций и т.д. Перечисленное хорошо согласуется с проблемами обучения, а качество обучения разработке алгоритмов определяется свойствами используемых лингвистических средств и наличием среды, обеспечивающей их исполнимость.

Соответственно качество обучения, его эффективность могут быть в значительной мере повышены использованием специальных компьютерных сред (подобных системам программирования, системам автоматизации проектирования), базирующимся на принципе прототипирования программ, что обеспечивает по аналогии с указанными средствами системность обучения, исполнимость, проверяемость разрабатываемых спецификаций.

Система обучения должна содержать [8]: – в части программного обеспечения средства прототипирования, средства разработки алгоритмов, управления базой данных, сервисные средства; – в части лингвистического обеспечения языки спецификаций, языковые средства проектирования алгоритмов [9]; – в части информационного обеспечения базы данных проектных решений.

Муравьёв Геннадий Леонидович, *к.т.н., профессор кафедры интеллектуальных информационных технологий Брестского государственного технического университета.*

Хвещук Владимир Иванович, *к.т.н., доцент, профессор кафедры интеллектуальных информационных технологий Брестского государственного технического университета.*

Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.

"Ядро" средств прототипирования должно обеспечивать разработку и редактирование спецификаций модулей, схем иерархии модулей, сценариев, прототипирование интерфейсов, генерацию исполнимых шаблонов модулей, специализированных каркасов (например, консольных и оконных каркасов windows-приложений с упрощенным и с полноценным графическим интерфейсом), тестирование прототипов, хранение, документирование и визуализацию результатов проектирования.

Важная часть такой среды – входной язык, лингвистические средства. Описания алгоритмов должны быть инвариантны к языкам программирования высокого уровня, читаемыми, исполнимыми, легко контролируемыми вручную и автоматически в процессе разработки. Описания в итоге должны быть готовым документом для автоматического или ручного кодирования на подходящем языке программирования. Соответственно входной язык должен напоминать естественный с возможностью описывать тексты по правилам, близким к правилам естественного языка, отвечать принципам структурного программирования, базироваться на минимальном наборе изобразительных средств, управляющих структур, отображающих соответствующие математические понятия и правила пользования ими. Используется процедурный механизм построения алгоритмов
<программа> ::= <основной_блок> <подпрограммы> ,
<подпрограмма> ::= <функция> | <процедура> .

Для повышения читаемости текстов наряду с традиционными строчными и фрагментарными комментариями дополнительно используются: – "встраиваемые" комментарии (комментарии, помещаемые в любое место текста, в команду без выделения спецсимволами); – гибкая префиксно-постфиксная система формирования идентификаторов переменных, названий функций и процедур, обеспечивающая возможность склонения используемых в описании алгоритмов идентификаторов в зависимости от контекста; – "подсветка синтаксиса" (выделение синтаксических единиц для удобства восприятия текстов алгоритмов при работе в редакторе текстов). При этом совместное использование указанных средств позволяет формулировать осмысленные тексты (команды) по правилам, напоминающим правила русского языка. "Встраиваемые" комментарии записываются строчными буквами кириллицы, тогда как управляющие конструкции прописными. Соответственно
<идентификатор>::=[<приставка>]<корень>[<окончание>]
(["_" [<приставка>] <корень> [<окончание>]]) ,
<корень> ::= <прописная_буква> ([<прописная_буква> | "_" | <цифра>]) ,
<приставка> ::= <встраиваемый_комментарий> ,
<окончание> ::= <встраиваемый_комментарий> ,
<встраиваемый_комментарий> ::= <строчная_буква> ([<строчная_буква >]) .

Инструментальные средства разработки алгоритмов должны обеспечивать: редактирование и сохранение алгоритмов, доступ к справочной информации; исполнимость кодов алгоритмов, управление, в том числе пошаговое, выполнением алгоритмов, контроль их корректности; генерацию программ на языках высокого уровня, документирование и визуализацию результатов; анализ и структурирование ЯВУ-кодов. Средства включают текстовый редактор, исполнитель, отладчик алгоритмов, генератор ЯВУ-текстов, систему управления файлами, окнами.

Исполнимость описаний алгоритмов изначально предполагает как формализованное построение языка, так и наличие исполнителя - соответствующей программной поддержки генерации загрузочных кодов либо интерпретации спецификаций алгоритмов. Поскольку интерпретация проще в реализации, а быстродействие не является критичным для такого рода систем, то предлагается использовать модификации указанных подходов.

Модификация интерпретационного подхода использует возможности существующих компиляторов и предполагает автоматическую трансформацию псевдопрограмм в адекватный в функциональном отношении ЯВУ-текст с последующим получением исполнимого кода. Формат внутреннего представления ЯВУ-текста базируется на теореме о структурировании программ и методе введения псевдопеременной. Это позволяет представить любую исходную спецификацию алгоритма в виде управляемой последовательности тактов выполнения. Для этого требуются: 1) правила трансформации; 2) конвертор спецификаций алгоритмов в ЯВУ-текст; 3) типовой транслятор ЯВУ; 4) монитор, управляющий исполнением спецификаций.

Модификация компиляционного подхода использует принцип виртуальной машины и предполагает автоматическую генерацию промежуточного кода, исполняемого (интерпретируемого) виртуальной машиной. Это упрощает реализацию системы, расширяет возможности диагностики и локализации ошибок. В качестве языка внутреннего представления спецификаций выбирается язык виртуальной машины, что позволяет представить любой исходный текст в виде исполнимых команд. Получение промежуточного кода предполагает выполнение традиционных этапов лексического, синтаксического, семантического анализа спецификаций. Для этого требуются: 1) правила генерации; 2) препроцессор исходных спецификаций; 3) генератор кода виртуальной машины; 4) виртуальная машина, составляющая ядро системы исполнения.

База данных и соответствующее ПО поддерживают работу с объектами, документами. Это спецификации данных, задач, функций, сценариев (диаграммы потоков данных, схемы данных, диаграммы прецедентов, описания потоков событий и др.), схемы программ, спецификации модулей, схемы иерархии модулей, спецификации интерфейсов (диаграммы объектов, состояний, робастности и др.), шаблоны модулей, сгенерированные каркасы. В качестве средств описаний могут использоваться, например, схемы ГОСТ 19.701-90, диаграммы UML.

Заключение. Развитие индустрии программных систем характеризуется значительным усложнением процесса разработки. Анализ тенденций позволяет сделать выводы о перспективности использования исполнимых спецификаций проектов (по аналогии с системами программирования) в том числе для обучения. В работе рассмотрены подходы и методы получения исполнимых кодов спецификаций алгоритмов, возможность применения виртуальной машины, структура соответствующих средств поддержки. Результаты макетировалась с использованием языков высокого уровня с ориентацией на ОС Windows и поддержкой интегрированной, многооконной, русифицированной среды пользователя. Рассмотренный подход обеспечивает автоматизацию индивидуального обучения разработке программ на единой методической и информационной основе, активизирует обучение в современных технологиях, начиная с этапа прототипирования. Кроме этого, пользователь-непрофессионал сможет решать задачи в содержательной форме без знания языка программирования.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Касьянов, В.Н. Проблемы обучения информатике и программированию / В.Н. Касьянов // Информационно-коммуникационные технологии в образовании (IST/IMS-2001) [Электронный ресурс]. – 2001. – Режим доступа: http://www.ict.edu.ru. – Дата доступа: 1.02.2010.
2. Лисков, Б. Использование абстракций и спецификаций при разработке программ / Б. Лисков, Дж. Гатэг. – М.: Мир, 1989. – 424 с.
3. Кузин, С.Г. Конструирование компонент программного обеспечения на основании графовых моделей [Электронный ресурс]. – 2004. – Режим доступа: http://www.unn.ru/vmk/graphmod.
4. Шалыто, А.А. SWITCH-технология: автоматный подход к созданию программного обеспечения "реактивных" систем / А.А. Шалыто, Н.И. Туккель // Программирование. – 2001. – № 5. – С. 45–62.
5. Леонов, А.Г. Программные исполнители в системе КуМир / А.Г. Леонов // Информационные технологии в образовании: материалы II междунар. конф. [Электронный ресурс]. – 2008. – Режим доступа: http://ito.edu.ru/sp/SP/SP-0-2007_11_13.html.
6. Кузин, С.Г. Компьютерная технология обучения основам алгоритмизации / С.Г. Кузин, Р.О. Митин, И.С. Скрибловский // [Электронный ресурс]. – 2008. – Режим доступа: www.unn.ru/vmk/graphmod.
7. Якименко, О.В. Применение программ-тренажеров в обучении программированию [Электронный ресурс]. – 2007. – Режим доступа: yakimenkoov@tspu.edu.ru.
8. Муравьев, Г.Л. Информационные технологии для обучения конструированию программ / Г.Л. Муравьев, В.И. Хвещук // Инновационные технологии обучения физико-математическим дисциплинам: материалы III междунар. научно-практ. Конференции. – Мозырь, 2011. – С. 156–157.
9. Муравьев, Г.Л. Автоматизация обучения алгоритмизации и программированию / Г.Л. Муравьев, С.В. Мухов // Вести ИСЗ. – 2004. – № 3. – С. 24–29.

Материал поступил в редакцию 14.11.11

MURAVJOV G.L., KHVIASHCHUK V.I. About creation of systems of training to construction of programs

Questions of the organization of systems of training to development of the programs which are based on principles of prototyping and feasibilities of specifications are considered. Requirements to characteristics of systems, the recommendation about creation linguistic and the software are led.

УДК 621.395.66

Ярошевич А.В.

СХЕМА КОМПЕНСАЦИИ РЕАКТИВНОЙ МОЩНОСТИ В КВАРТИРНЫХ ЭЛЕКТРИЧЕСКИХ СЕТЯХ

Электрические цепи бытового потребителя питаются, как правило, одной фазой трёхфазной цепи переменного тока с нулевым проводом. Традиционно нагрузка в таких цепях считается резистивной, и учёт потреблённой электроэнергии производится однофазным индукционным счётчиком, учитывающим потребляемую энергию активной мощности (АМ) нагрузки.

В настоящее время характер нагрузки в цепях бытовых потребителей существенно изменился. Распространение бытовых приборов с трансформаторами, электродвигателями и сложными электронными цепями привело к появлению реактивной (индуктивной) составляющей мощности. По данным [1], cos φ в таких цепях может составлять cos φ = 0,65 … 0,97. Средневзвешенное значение cos φ = 0,85 [2].

Наиболее удачным показателем, характеризующим величину потребления реактивной мощности (РМ), является коэффициент РМ tg φ = *Q*/*P*, где *Q*, *P* – соответственно величины РМ и АМ. Передача РМ к потребителю и ее потребление в сети приводят к дополнительным потерям АМ в распределительных электрических сетях. При значении cos φ = 0,85 РМ составляет 60% от АМ. Вследствие этого возрастают и активные потери от передачи электроэнергии, которые при *cos* φ = 0,85 составляют 15% от полезной активной мощности у потребителя при 10% в случае чисто активной нагрузки.

Распределение доли подключённой РМ неравномерно по часам суток и индивидуально для каждого потребителя.

Принимая меры для полного учёта распределяемой энергии, поставщики переходят к установке электронных счётчиков, обеспечивающих учёт полной потребляемой энергии. В таких условиях у потребителя имеется возможность сократить потреблённую энергию за счёт известного принципа компенсации индуктивной мощности подключением дополнительной ёмкостной нагрузки [3].

Существующие схемы и устройства компенсации РМ созданы для мощных промышленных потребителей и не могут использоваться в квартирных распределительных сетях.

Наглядное представление о сущности компенсации реактивной мощности даёт рис. 1. На рис. 1а) изображена схема электрической цепи. Пусть до компенсации потребитель имел активную мощность P, соответственно ток I_a (отрезок ***OB*** на рис 1б) и реактивную мощность от индуктивной нагрузки Q_L с соответствующим током I_L (отрезок ***BA***). Полной мощности S_1 соответствует вектор I_H (отрезок ***OA***). Коэффициент мощности до компенсации cos φ_1.

Векторная диаграмма компенсации представлена на рис. 1в). После компенсации, т.е. после подключения параллельно нагрузке конденсаторной установки ***КУ*** с мощностью Q_K (ток I_C), суммарная реактивная мощность потребителя будет уже Q_1–Q_K (ток I_L–I_C) и соответственно снизится угол сдвига фаз с φ_1 до φ_2 и повысится коэффициент мощности с cos φ_1 до cos φ_2. Полная потребляемая мощность при той же потребляемой активной мощности P (токе I_a) снизится с S_1 (ток I_H) до S_2 (ток I_2) (отрезок ***OA'***). Следовательно, в результате компенсации можно при том же сечении проводов повысить пропускную способность сети.

За счёт присоединения к сети ***КУ*** с мощностью Q_K уменьшаются потери мощности. После компенсации потери мощности

$$\Delta P = \frac{\left|P^2 + \left(Q - Q_K\right)^2\right| R}{U_{НОМ}^2} + \Delta P_{КУ},$$

где $\Delta P_{КУ}$ – потери мощности в компенсирующем устройстве, кВт.

При реализации схемы компенсации РМ необходимо решить следующие проблемы:

- измерение величины реактивных потерь в реальном времени;
- определение ёмкости компенсирующего конденсатора;
- коммутация к сети ёмкости из конденсаторной батареи.

Общая схема предлагаемого устройства для компенсации РМ бытового потребителя электрической энергии представлена на рис. 2.

Схема компенсации РМ (***СКРМ***) подключается между счётчиком электроэнергии и нагрузкой потребителя. Основными узлами ***СКРМ*** являются:

- схема измерения разности фаз сетевого напряжения и потребляемого тока, отражающей долю РМ в нагрузке;
- схема управления тиристорными ключами для коммутации компенсирующей ёмкости;

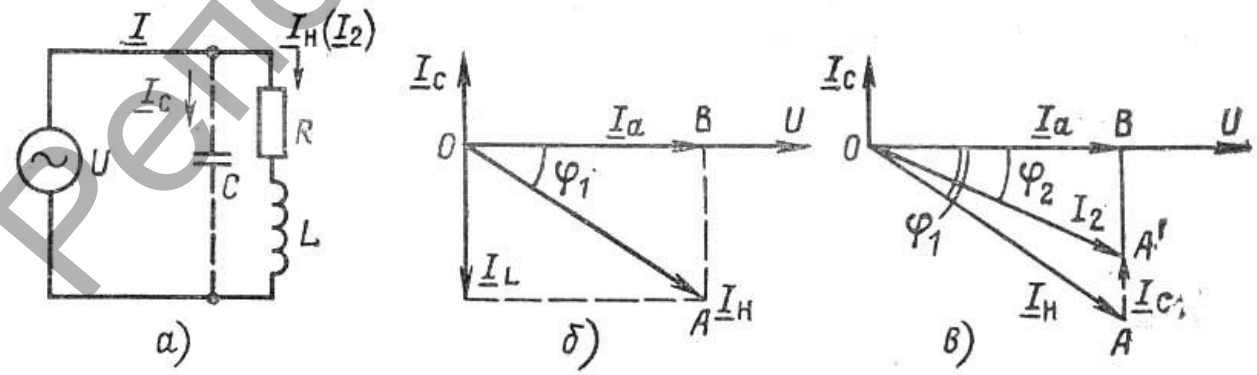

Рис. 1. Векторная диаграмма компенсации реактивной мощности

Ярошевич Анатолий Васильевич*, доцент кафедры автоматизации технологических процессов и производств Брестского государственного технического университета.*

Беларусь, БрГТУ, 224017, г. Брест, ул. Московская, 267.